

THIS OPINION WAS NOT WRITTEN FOR PUBLICATION

The opinion in support of the decision being entered today (1) was not written for publication in a law journal and (2) is not binding precedent of the Board.

Paper No. 24

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JAMES A. KAHLE
and CHIN-CHENG KAU

Appeal No. 96-1426
Application 08/001,863¹

ON BRIEF

Before BARRETT, TORCZON, and CARMICHAEL, Administrative Patent Judges.

BARRETT, Administrative Patent Judge.

¹ Application for patent filed January 8, 1993, entitled "Method And System For Increased Instruction Synchronization Efficiency In A Superscalar Processor System Utilizing Partial Data Dependency Interlocking."

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134 from the rejection of claims 1, 3, 6, and 7, which constitute all the claims pending in the application. Claims 2, 4, and 5 have been cancelled.

The invention is directed to a method and system for eliminating data dependency hazards in a superscalar computer which includes instructions having a greater number of source operands than may be interlocked utilizing a data dependency interlock circuit. Data dependency hazards occur, for example, when a first instruction must write its result (to a destination operand) before the second instruction can read and subsequently use the result (as a source operand). To allow this write before read, execution of the read must be blocked until the write has occurred; that is, there is an "interlock" between the write and read instructions, specifically between the destination operand and the source operand. Modern superscalar machines use "data dependency interlock circuits" which "contain logic which operates in concert with instruction dispatch circuitry to ensure that an instruction is not dispatched until such time as a result from a preceding instruction which is necessary for correct execution of that instruction has been obtained" (specification, page 3). It is common in existing superscalar machines to use a source-to-destination interlock circuit "which is capable of interlocking two source operands with two destination operands, to ensure that data dependency hazards for such operands clear prior to permitting dispatch of an instruction containing these operands" (specification, page 9). To use an instruction with three operands would require a proportional increase in the logic required to implement a data

dependency interlock circuit (specification, page 3). Appellants' method and system eliminates data dependency hazards where the instructions include a greater number of source operands than may be interlocked utilizing existing data dependency interlock circuitry. If the source-to-destination dependency interlock circuitry can interlock N source operands, an instruction having more than N source operands is dispatched only upon elimination of N possible data dependency hazards by the source-to-destination dependency interlock circuitry and a determination that all instructions preceding the remaining source operands have completed.

Claim 1 is reproduced below.

1. A method for increased efficiency in instruction synchronization in a superscalar processor system capable of simultaneously dispatching multiple scalar instructions having multiple source and destination operands and having source-to-destination dependency interlock circuitry capable of interlocking N source operands and M destination operands to prevent data dependency hazards, said method comprising the steps of:

dispatching each scalar instruction within said superscalar processor system which includes no more than N source operands upon elimination of possible data dependency hazards, as indicated by said source-to-destination dependency interlock circuitry; and

dispatching each scalar instruction which includes more than N source operands only upon elimination of possible data dependency hazards for a first N source operands, as indicated by said source-to-destination dependency interlock circuitry and a completion of all preceding instructions, wherein possible data dependencies for scalar instructions which include N+1 or more source operands are avoided without requiring additional dependency interlock circuitry.

The examiner relies on the following reference:

Vassiliadis et al. (Vassiliadis) 5,051,940 Sep. 24, 1991

Claims 1, 3, 6, and 7 stand rejected under 35 U.S.C. § 102(a) as being anticipated by Vassiliadis.

OPINION

We reverse.

"Anticipation is established only when a single prior art reference discloses, expressly or under principles of inherency, each and every element of a claimed invention." RCA Corp. v. Applied Digital Data Systems, Inc., 730 F.2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984).

Vassiliadis discloses "a data dependency collapsing arithmetic logic unit (ALU)," which is the special three-to-one (three-operand, single-result) ALU of figure 9 to eliminate data hazard interlocks using compounded instructions. By "collapses data dependency" it is meant "that a pair of instructions can be executed even when one of the pair requires as an operand the result produced by execution of the other of the pair of instructions" (column 3, lines 45-48). "The term 'compounding' refers to grouping of instructions contained in a sequence of instructions, the grouping being for the purpose of concurrent or parallel execution of the grouped instructions. At minimum, compounding is represented by 'pairing' of two instructions for simultaneous execution." (Column 5, lines 13-19.) "The proposed apparatus of this invention collapses these interlocks by deriving new functions that arise from combining the execution of instructions whose operands present data hazards while retaining the execution of functions inherent in the instruction set" (column 8, lines 25-30).

Consider the sequence of instructions, where NR and AR are opcodes and R1-R4 are registers (note that these instructions have the form: "Opcode (Destination), (Source)") (column 9, line 68 to column 10, line 1):

NR R1, R2
AR R3, R4

Data dependency interlocks occur with register combinations 7-12 listed at the top of column 11, e.g., "R1=R3+R2+R4" (i.e., destination operand R1 must be written by instruction NR before it is written by instruction AR) or "R1=R4+R2+R3" (i.e., destination operand R1 must be written by instruction NR before it is read as a source operand by instruction AR). Vassiliadis considers two types of instructions: logical (LOGICAL) and arithmetic (ADD). The four sequences of the two types of operations (LOGICAL/LOGICAL, LOGICAL/ADD, ADD/ADD, ADD/LOGICAL) give rise to data dependency interlocks for each of the five interlock conditions, for a total of 20 instruction combinations with interlocking. Figure 5 specifies the operations that must be performed on the operands to collapse the interlocks. Figures 6A and 6B specify the ALU operations required to be performed on operand inputs AI0, AI1, and AI2. Figures 7A and 7B specify the routing of the operands for the ALU to perform the operations of figure 5. Figure 9 shows a logical representation of the data dependency collapsing

ALU, where the inputs AI0, AI1, and AI2 are shown in figures 7A, 7B.

Vassiliadis describes that "[a]ddress generation can also be affected by data hazards which will be referred to as address hazards, AHAZ" (column 15, approx. lines 42-44). Address instructions can have two source operands, e.g., R1 and R5 in the instruction S R3, D(R1,R5) (column 15, line 55). The data dependencies in pairs of instructions which include an address instruction can be collapsed following similar procedures to those described above for pairs of logical and arithmetic instructions. Vassiliadis states (column 15, line 64 to column 16, line 2):

For an interlock collapsing ALU, new operations arising from collapsing AHAZ interlocks must be derived by analyzing all combinations of instruction sequences and address operand conflicts. Analysis indicates that common interlocks, such as the ones contained in the above instruction sequences, can be collapsed with a four-to-one ALU.

Vassiliadis also discloses a branch hazard-collapsing ALU (column 17, lines 1-45).

It is not clear whether the examiner believes that Vassiliadis is doing the same thing as the claimed invention or only finds that the claims, as presented, happen to read on Vassiliadis. To start with, the examiner finds that "Vassiliadis

taught the invention as claimed including a data dependency collapsing hardware system for scheduling" (Examiner's Answer, page 2). The examiner does not explain what the "collapsing hardware" has to do with the claimed subject matter and apparently misapprehends how Vassiliadis works. While Vassiliadis does address the problem of data dependency interlocks in superscalar machines, the approach is completely different than that recited in the claims. Vassiliadis eliminates interlocks between two instructions by collapsing the data dependency between the instructions by concurrently executing the instructions using a special ALU having provision for receiving three operands which are used by the first and second instructions or, for address hazards, using an ALU having four operand inputs. The data dependency collapsing ALU in Vassiliadis is not "source-to-destination dependency interlock circuitry" as claimed because it does not interlock instructions, i.e., it does not block or delay dispatch of an instruction until a result from a preceding instruction which is necessary for correct execution has been obtained, causing the instructions to execute serially. The collapsing ALU eliminates interlocks between instructions by concurrent execution of pairs of instructions which have possible data dependencies using a

special ALU. Therefore, it is difficult to see the relevance of Vassiliadis.

As to the limitation of "dispatching each scalar instruction . . . which includes no more than N source operands upon elimination of possible data dependency hazards, as indicated by said source-to-destination dependency interlock circuitry," the examiner finds (Examiner's Answer, pages 2-3):

a)each scalar instruction [NR R1,R2] or [AR R3,R4] (e.g. see col.9, lines 65-68; col.10, lines 2-5) which includes no more than N (e.g. (R1,R2) or (R3,R4); N =2) source operands upon eliminating (e.g. see col.9, lines 45-56) possible [sic, possible] data hazards by a dependency interlock circuit [ALU] (e.g. see col.10, lines 12-68; col.11, lines 1-16; see also col.2, lines 58-68 for the relief of data dependency interlocks);

The logical and arithmetic instructions have only one source operand, the other operand is the destination operand, so N=1, not N=2 as found by the examiner. The data dependency collapsing ALU in Vassiliadis is not "source-to-destination dependency interlock circuitry" because it does not interlock instructions: it eliminates interlocks between instructions using a special ALU. However, Vassiliadis discloses in the background that the detection and resolution of structural and data dependency hazards may be implemented in hardware (column 1, lines 51-55). This presumably is the same hardware as the "data dependency interlock circuits" in the prior art described by appellants

(e.g., specification, page 3) and the "source-to-destination dependency interlock circuitry" in claim 1. Vassiliadis's invention does not deal with this interlock hardware, nor does the examiner rely on the hardware in the background of the invention. Nevertheless, because such hardware for interlocking N source operands is described in the background of Vassiliadis and is admitted by appellants to be well known (specification, page 3), we find that the limitation of "dispatching each scalar instruction . . . which includes no more than N source operands upon elimination of possible data dependency hazards, as indicated by said source-to-destination dependency interlock circuitry" to be disclosed.

As to the limitation of "dispatching each scalar instruction which includes more than N source operands only upon elimination of possible data dependency hazards for a first N source operands, as indicated by said source-to-destination dependency interlock circuitry and a completion of all preceding instructions," the examiner finds (Examiner's Answer, page 3):

b)each scalar instruction (e.g. S r3,D(R4,R5) in col.16, line 36; see also S R3,D(R4,R1) in (5) and S R3,D(R1,R5) in (4); see fig.2 for scalar instructions) which includes more than N source operands (R4,R5,D) only upon elimination of possible hazards for first N sources (r4,r5) (e.g. see col.16, lines 37-65; see the detection of r4 or r5 with r1) as detected by the interlock circuitry [ALU], wherein possible data dependencies for scalar

instructions (e.g. see NR r1,r2 and S r3,D(R4,R5) in col.16, lines 35-36; see also the AGI0-3 of the source operands) which include more than N source operands were collapsed (e.g. see col.15, lines 64-68; col.16, lines 1-9) using the interlock collapsing ALU (see also branch hazard-collapsing ALU for instructions BCT and AL which had more than N operands in col.17, lines 24-27).

The examiner errs in finding this limitation to be disclosed. First, the data dependency collapsing ALU in Vassiliadis is not "source-to-destination dependency interlock circuitry" because it does not interlock instructions: it eliminates interlocks between instructions. Thus, Vassiliadis does not disclose "elimination of possible data dependency hazards for a first N source operands, as indicated by said source-to-destination dependency interlock circuitry" for an "instruction which includes more than N source operands." Second, Vassiliadis does not disclose instructions having more source operands than can be interlocked with "source-to-destination dependency interlock circuitry." The data dependency collapsing ALU is not interlock circuitry and it eliminates all data dependencies between two instructions, not just some. Thus, Vassiliadis does not disclose eliminating data dependency hazards for N source operands with "source-to-destination dependency interlock circuitry" and thereafter awaiting execution of all previous instructions prior to execution of the remaining instruction operands. Third, the

examiner's reliance on the address hazard-collapsing ALU in Vassiliadis as teaching N+1 operands is misplaced. It is true that the address instructions are shown with two source operands instead of one for the logical and arithmetic instructions. However, Vassiliadis handles sequences with address instructions having two source operands in the same manner as logical and arithmetic instructions having one operand: collapsing the data dependencies using a special ALU, in this case one with four inputs. Nothing in Vassiliadis suggests eliminating some data dependencies for operands with interlock hardware and then awaiting execution of all previous instructions prior to the remaining instruction operands.

It is admitted that "[o]ne technique for ensuring that such so-called 'data dependency hazards' do not occur is the restriction of the dispatching of a particular instruction until such time as all preceding instructions have been dispatched" (specification, page 3). It is stated that "the method and system of the present invention processes three source operand instructions by delaying dispatch of such instructions until all preceding instructions have completed, eliminating possible data dependency hazards" (specification, page 10). Thus, the condition of delaying dispatch until all preceding instructions

have completed appears to be the same. What we do not find is appellants' partial data dependency interlocking step of using "source-to-destination dependency interlock circuitry" to eliminate N possible data dependency hazards and then determining that all previous instructions have completed before dispatching the instruction.

For the reasons discussed above, the rejection of claims 1, 3, 6, and 7 is reversed.

REVERSED

LEE E. BARRETT)	
Administrative Patent Judge)	
)	
)	
)	
)	BOARD OF PATENT
RICHARD TORCZON)	APPEALS
Administrative Patent Judge)	AND
)	INTERFERENCES
)	
)	
)	
JAMES T. CARMICHAEL)	
Administrative Patent Judge)	

Appeal No. 96-1426
Application 08/001,863

Andrew J. Dillon
FELSMAN, BRADLEY, GUNTER & DILLON, LLP
Suite 350, Arboretum Point
9505 Arboretum Boulevard
Austin, TX 78759